IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application. No: 10/021,535	§	
Filed: December 12, 2001	§	Atty. Dkt. No: 6002-07000
Inventor(s): Anuszczyk, et al.	§	
Title: METHOD AND APPARATUS FOR MANAGING COMPONENTS IN AN IT SYSTEM	§	
Examiner: Shaw, Peling Andy	§	
Group/Art Unit: 2144	§	

APPEAL BRIEF

Dear Sir/Madam:

Further to the Notice of Appeal filed April 10, 2008, Appellant presents this Appeal Brief. Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by Symantec Operating Corporation, a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 20330 Stevens Creek Blvd, Cupertino, CA, 95014.

II. RELATED APPEALS AND INTERFERENCES

No related appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1, 19-21, 25-29, 31, and 34 are canceled.

Claims 2-18, 22-24, 30, 32-33, and 35-40 are pending in the application. All of the pending claims stand rejected and are the subject of this appeal. A copy of the claims incorporating entered amendments is included in the Claims Appendix hereto.

IV. STATUS OF AMENDMENTS

All amendments have been entered. The Claims Appendix hereto reflects the current state of the claims.

V. SUMMARY OF THE INDEPENDENT CLAIMS

Claim 5 recites a method comprising creating a plurality of component fingerprints, including a fingerprint for a first component. (*See, e.g., block 200 of FIG. 9; p. 30, lines 6-7*). Creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component. (*See, e.g., p. 15, lines 24-29; p. 16, lines 14-24; and p. 18, lines 24-28*).

The method further comprises automatically discovering the existence of a plurality of components in an information technology (IT) system (*see, e.g., p. 6, lines 27-28*) using the plurality of component fingerprints, where the discovered components include the first component. (*See, e.g., block 204 of FIG.9; p. 30, lines 11-12*). Automatically discovering the existence of the first component comprises: 1) Receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system (*see, e.g., p. 8, lines 20-24; p. 9, lines 3-5; p. 16, lines 4-9*), where each event message matches a respective attribute of the fingerprint for the first component (*see, e.g., p. 14, lines 10-15; p. 16, lines 4-9 and lines 21-22; p. 30, lines 15-17*); and 2) determining that event messages matching every attribute of the fingerprint for the first component have been received (*see, e.g., p. 14, lines 10-15; p. 16, lines 8-9 and 23-24; p. 30, lines 15-18*).

The method further comprises automatically determining at least one dependency between two or more of the discovered components (*see, e.g., block 206 of FIG. 9; p. 31, line 1*), and tracking changes to at least one of the discovered components and the at least one dependency between two or more of the discovered components (*see, e.g., block 208 of FIG. 9; p. 31, lines 2-3*).

Claim 13 recites a computer-readable medium storing program instructions. (*E.g., the program instructions of the observer service 50 and analysis service 70 of FIG. 2*). The program instructions are executable to implement a method comprising identifying a plurality of attributes of a first component and listing one or more, but not

all, of the plurality of attributes in a fingerprint for the first component. (*See, e.g., p. 15, lines 24-29; p. 16, lines 14-24; and p. 18, lines 24-28*).

The method implemented by the program instructions further comprises an observer module (*see, e.g., observer service 50 of FIG. 2*) detecting a plurality of real-time events (*see, e.g., p. 8, lines 20-24; p. 9, lines 3-5; p. 16, lines 4-9*) in an information technology (IT) system (*see, e.g., p. 6, lines 27-28*). Each event matches a respective attribute listed in the fingerprint for the first component. (*See, e.g., p. 14, lines 10-15; p. 16, lines 4-9 and lines 21-22; p. 30, lines 15-17*).

The method implemented by the program instructions further comprises the observer module sending a respective event message to an analysis module (*see, e.g., analysis service 70 of FIG. 2*) in response to each of the plurality of real-time events (*see, e.g., p. 8, lines 20-24; p. 9, lines 3-5; p. 16, lines 4-9*).

The method implemented by the program instructions further comprises the analysis module accumulating the event messages (*see, e.g., accumulator 80 of FIG. 2; p. 14, lines 10-15*) and analyzing the event messages in order to determine that events matching every attribute listed in the fingerprint for the first component have occurred in the IT system (*see, e.g., p. 14, lines 10-15; p. 16, lines 8-9 and 23-24; p. 30, lines 15-18*).

The method implemented by the program instructions further comprises the analysis module indicating that the first component exists in the IT system in response to determining that events matching every attribute listed in the fingerprint for the first component have occurred in the IT system. (*See, e.g., p. 14, lines 10-15; p. 16, lines 8-9 and 23-24; p. 30, lines 15-18*).

Claim 14 recites a system comprising means (*see, e.g., fingerprint creation module 124 of FIG. 3*) for creating a plurality of component fingerprints, including a fingerprint for a first component. (*See, e.g., block 200 of FIG. 9; p. 30, lines 6-7*). Creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of

attributes as the fingerprint for the first component. (*See, e.g., p. 15, lines 24-29; p. 16, lines 14-24; and p. 18, lines 24-28*).

The system further comprises means (*see, e.g., observer service 50 and analysis service 70 of FIG. 2*) for automatically discovering the existence of a plurality of components in an information technology (IT) system (*see, e.g., p. 6, lines 27-28*) using the plurality of component fingerprints, where the discovered components include the first component. (*See, e.g., block 204 of FIG.9; p. 30, lines 11-12*). Automatically discovering the existence of the first component comprises: 1) Receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system (*see, e.g., p. 8, lines 20-24; p. 9, lines 3-5; p. 16, lines 4-9*), where each event message matches a respective attribute of the fingerprint for the first component (*see, e.g., p. 14, lines 10-15; p. 16, lines 4-9 and lines 21-22; p. 30, lines 15-17*); and 2) determining that event messages matching every attribute of the fingerprint for the first component have been received (*see, e.g., p. 14, lines 10-15; p. 16, lines 8-9 and 23-24; p. 30, lines 15-18*).

The system further comprises means (*see, e.g., rule engine 74 and dependency detection rules 78 of FIG. 2*) for automatically determining at least one dependency between two or more of the discovered components (*see, e.g., block 206 of FIG. 9; p. 31, line 1*) and means (*see, e.g., rule engine 74 of FIG. 2*) for tracking changes to at least one of the discovered components and the at least one dependency between two or more of the discovered components (*see, e.g., block 208 of FIG. 9; p. 31, lines 2-3*).

Claim 15 recites an apparatus comprising a memory storing program instructions and a processor in communication with the memory. (*See, e.g., server 11 of FIG. 1, which includes memory storing program instructions of the agent 12, and a processor in communication with the memory; p. 7, lines 29-30; p. 8, lines 9-10*). The processor is directed by the program instructions to create a plurality of component fingerprints, including a fingerprint for a first component. (*See, e.g., block 200 of FIG. 9; p. 30, lines 6-7*). Creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of

attributes as the fingerprint for the first component. (*See, e.g., p. 15, lines 24-29; p. 16, lines 14-24; and p. 18, lines 24-28*).

The processor is further directed by the program instructions to automatically discover the existence of a plurality of components in an information technology (IT) system (*see, e.g., p. 6, lines 27-28*) using the plurality of component fingerprints, where the discovered components include the first component. (*See, e.g., block 204 of FIG.9; p. 30, lines 11-12*). Automatically discovering the existence of the first component comprises: 1) Receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system (*see, e.g., p. 8, lines 20-24; p. 9, lines 3-5; p. 16, lines 4-9*), where each event message matches a respective attribute of the fingerprint for the first component (*see, e.g., p. 14, lines 10-15; p. 16, lines 4-9 and lines 21-22; p. 30, lines 15-17*); and 2) determining that event messages matching every attribute of the fingerprint for the first component have been received (*see, e.g., p. 14, lines 10-15; p. 16, lines 8-9 and 23-24; p. 30, lines 15-18*).

The processor is further directed by the program instructions to automatically determine at least one dependency between two or more of the discovered components (*see, e.g., block 206 of FIG. 9; p. 31, line 1*) and to track changes to at least one of the discovered components and the at least one dependency between two or more of the discovered components (*see, e.g., block 208 of FIG. 9; p. 31, lines 2-3*).

Claim 16 recites a method comprising creating a fingerprint for a first component, where creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component. (*See, e.g., block 200 of FIG. 9; p. 30, lines 6-7; p. 15, lines 24-29; p. 16, lines 14-24; and p. 18, lines 24-28*).

The method further comprises creating a subfingerprint for a refinement of the first component, where the subfingerprint for the refinement of the first component includes one or more attributes of the refinement of the first component. (*See, e.g., p. 2, lines 16-19; p. 19, lines 24-30*).

The method further comprises automatically discovering the first component in an information technology (IT) system. (*See, e.g., p. 6, lines 27-28*). Automatically discovering the first component comprises: 1) receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system (*see, e.g., p. 8, lines 20-24; p. 9, lines 3-5; p. 16, lines 4-9*), where each event message matches a respective attribute of the fingerprint for the first component (*see, e.g., p. 14, lines 10-15; p. 16, lines 4-9 and lines 21-22; p. 30, lines 15-17*); and 2) determining that event messages matching every attribute of the fingerprint for the first component have been received (*see, e.g., p. 14, lines 10-15; p. 16, lines 8-9 and 23-24; p. 30, lines 15-18*).

The method further comprises, in response to discovering the first component, performing one or more commands to obtain information regarding the first component. (*See, e.g., p. 20, lines 7-11 and 25-26; p. 33, lines 2-6*). The method further comprises automatically discovering the refinement of the first component in the IT system by matching the information regarding the first component to the one or more attributes of the refinement of the first component included in the subfingerprint for the refinement of the first component. (*See, e.g., p. 33, lines 8-10; p. 20, lines 7-11 and 28-30; p. 21, lines 4-6; p. 30, lines 18-23*).

Claim 23 recites a computer-readable medium storing instructions that direct a microprocessor. (*E.g., the program instructions of the observer service 50 and analysis service 70 of FIG. 2*). The instructions direct the microprocessor to create a fingerprint for a first component, where creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component. (*See, e.g., block 200 of FIG. 9; p. 30, lines 6-7*; *p. 15, lines 24-29; p. 16, lines 14-24; and p. 18, lines 24-28*).

The instructions further direct the microprocessor to create subfingerprint for a refinement of the first component, where the subfingerprint for the refinement of the first component includes one or more attributes of the refinement of the first component. (*See, e.g., p. 2, lines 16-19; p. 19, lines 24-30*).

The instructions further direct the microprocessor to automatically discover the first component in an information technology (IT) system. (*See, e.g., p. 6, lines 27-28*). Automatically discovering the first component comprises: 1) receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system (*see, e.g., p. 8, lines 20-24; p. 9, lines 3-5; p. 16, lines 4-9*), where each event message matches a respective attribute of the fingerprint for the first component (*see, e.g., p. 14, lines 10-15; p. 16, lines 4-9 and lines 21-22; p. 30, lines 15-17*); and 2) determining that event messages matching every attribute of the fingerprint for the first component have been received (*see, e.g., p. 14, lines 10-15; p. 16, lines 8-9 and 23-24; p. 30, lines 15-18*).

The instructions further direct the microprocessor to, in response to discovering the first component, perform one or more commands to obtain information regarding the first component. (*See, e.g., p. 20, lines 7-11 and 25-26; p. 33, lines 2-6*). The instructions further direct the microprocessor to automatically discover the refinement of the first component in the IT system by matching the information regarding the first component to the one or more attributes of the refinement of the first component included in the subfingerprint for the refinement of the first component. (*See, e.g., p. 33, lines 8-10; p. 20, lines 7-11 and 28-30; p. 21, lines 4-6; p. 30, lines 18-23*).

Claim 24 recites an apparatus comprising a memory storing program instructions and a processor in communication with the memory. (*See, e.g., server 11 of FIG. 1, which includes memory storing program instructions of the agent 12, and a processor in communication with the memory; p. 7, lines 29-30; p. 8, lines 9-10*). The processor is directed by the program instructions to create a fingerprint for a first component, where creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component. (*See, e.g., block 200 of FIG. 9; p. 30, lines 6-7; p. 15, lines 24-29; p. 16, lines 14-24; and p. 18, lines 24-28*).

The processor is further directed by the program instructions to create subfingerprint for a refinement of the first component, where the subfingerprint for the refinement of the first component includes one or more attributes of the refinement of the first component. (*See, e.g., p. 2, lines 16-19; p. 19, lines 24-30*).

The processor is further directed by the program instructions to automatically discover the first component in an information technology (IT) system. (*See, e.g., p. 6, lines 27-28*). Automatically discovering the first component comprises: 1) receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system (*see, e.g., p. 8, lines 20-24; p. 9, lines 3-5; p. 16, lines 4-9*), where each event message matches a respective attribute of the fingerprint for the first component (*see, e.g., p. 14, lines 10-15; p. 16, lines 4-9 and lines 21-22; p. 30, lines 15-17*); and 2) determining that event messages matching every attribute of the fingerprint for the first component have been received (*see, e.g., p. 14, lines 10-15; p. 16, lines 8-9 and 23-24; p. 30, lines 15-18*).

The processor is further directed by the program instructions to, in response to discovering the first component, perform one or more commands to obtain information regarding the first component. (*See, e.g., p. 20, lines 7-11 and 25-26; p. 33, lines 2-6*). The instructions further direct the microprocessor to automatically discover the refinement of the first component in the IT system by matching the information regarding the first component to the one or more attributes of the refinement of the first component included in the subfingerprint for the refinement of the first component. (*See, e.g., p. 33, lines 8-10; p. 20, lines 7-11 and 28-30; p. 21, lines 4-6; p. 30, lines 18-23*).

Claim 30 recites a method for determining dependencies between components in an information technology (IT) system (*see, e.g., p. 6, lines 27-28*). The method comprises automatically discovering a first component and a second component in the IT system. (*See, e.g., block 204 of FIG.9; p. 30, lines 11-12*). Automatically discovering the first component comprises automatically discovering that one or more elements of the first component are present in the IT system, and automatically discovering the second component comprises automatically discovering that one or more elements of the second component are present in the IT system. (*See, e.g., p. 14, lines 10-15; p. 16, lines 8-9 and 23-24; p. 30, lines 15-18*).

The method further comprises monitoring the usage of resources by the discovered first and second components in the IT system by receiving real-time messages. (*See, e.g., p. 24, lines 17-25*).

In response to receiving a first real-time message indicating that the first component uses a particular resource, a first resource usage message is sent to an accumulator, where the first resource usage message indicates that the first component uses the particular resource. (*See, e.g., p. 25, lines 19-22*). Similarly, in response to receiving a second real-time message indicating that the second component uses the particular resource, a second resource usage message is sent to the accumulator, where the second resource usage message indicates that the second component uses the particular resource (*See, e.g., p. 25, lines 19-22*).

The method further comprises the accumulator indicating that a first dependency between the first component and the second component exists in response to determining that the first and second resource usage messages indicate that the first component and the second component both use the particular resource. (*See, e.g., p.26, lines 5-18 and lines 20-21*).

The method further comprises determining a type of the first dependency between the first component and the second component. (*See, e.g., p. 22, lines 3-4, line 19, and line 28; p. 26, lines 21-23*).

Claim 40 recites an apparatus comprising a memory storing program instructions and a processor in communication with the memory. (*See, e.g., server 11 of FIG. 1, which includes memory storing program instructions of the agent 12, and a processor in communication with the memory; p. 7, lines 29-30; p. 8, lines 9-10*). The processor is directed by the program instructions to implement a method comprising automatically discovering a first component and a second component in an information technology (IT) system. (*See, e.g., block 204 of FIG.9; p. 30, lines 11-12*). Automatically discovering the first component comprises automatically discovering that one or more elements of the first component are present in the IT system, and automatically discovering the second component comprises automatically discovering that one or more

elements of the second component are present in the IT system. (*See, e.g., p. 14, lines 10-15; p. 16, lines 8-9 and 23-24; p. 30, lines 15-18*).

The method implemented by the program instructions directing the processor further comprises monitoring the usage of resources by the discovered first and second components in the IT system by receiving real-time messages. (*See, e.g., p. 24, lines 17-25*).

In response to receiving a first real-time message indicating that the first component uses a particular resource, a first resource usage message is sent to an accumulator, where the first resource usage message indicates that the first component uses the particular resource. (*See, e.g., p. 25, lines 19-22*). Similarly, in response to receiving a second real-time message indicating that the second component uses the particular resource, a second resource usage message is sent to the accumulator, where the second resource usage message indicates that the second component uses the particular resource (*See, e.g., p. 25, lines 19-22*).

The method implemented by the program instructions directing the processor further comprises the accumulator indicating that a first dependency between the first component and the second component exists in response to determining that the first and second resource usage messages indicate that the first component and the second component both use the particular resource. (*See, e.g., p.26, lines 5-18 and lines 20-21*).

The method implemented by the program instructions directing the processor further comprises determining a type of the first dependency between the first component and the second component. (*See, e.g., p. 22, lines 3-4, line 19, and line 28; p. 26, lines 21-23*).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1) Claims 2-18 and 22-24 stand rejected under 35 U.S.C. 112, first paragraph.

2) Claims 2-5 and 11-15 stand rejected under 35 U.S.C. 102(a) as being anticipated by Kar et al., "An Architecture for Managing Application Services over Global Networks" (hereinafter "Kar").

3) Claims 30, 32-33, and 35-40 stand rejected under 35 U.S.C. 102(a) as being anticipated by Keller et al., "Dynamic Dependencies in Application Service Management" (hereinafter "Keller").

4) Claims 6-10 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Kar in view of Kathrow et al., U.S. Patent No. 6,393,438 B1 (hereinafter "Kathrow").

5) Claims 16-18 and 22-24 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Kar in view of Kathrow.

VII. ARGUMENT

1) Section 112 Rejections

Claims 2-18 and 22-24 stand rejected under 35 U.S.C. 112, first paragraph. Appellant respectfully traverses this rejection.

Claims 5, 14, and 15

With respect to **claims 5, 14, and 15**, the Examiner asserts:

> Independent claims 5 and 14-15 have been amended with the limitations of ". . .creating a plurality of component fingerprints . . . ", ". . . selecting one or more, but not all, of the plurality of attributes . . .", ". . .automatically discovering . . . using the plurality of component fingerprints . . . ", ". . .wherein each event message matches a respective attribute of the fingerprint . . ." and ". . .determining that event messages matching every attribute of the fingerprint . . ." that are not clearly specified in applicant's original specification or claim language. It would cause undue experimentation to one of ordinary skill in the art to make Applicant's invention.

Appellant disagrees and submits that there is very clear support for these limitations in the specification.

With respect to the claim limitations of, "creating a plurality of component fingerprints…" and "…wherein creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component…", see at least the following portions of the specification:

p. 15, lines 24-29:

> In a model-based discovery method, a model is constructed that defines a component, such as an application, and all of its component items, such as files and registry keys. The model, therefore, is a collection of data that defines the presence and attributes of the elements of an application or component. Using the model, a matching set that describes key elements of the application can be generated. This matching set, which can also be called a fingerprint, is a **subset** of the model for the application or component that uniquely identifies it…

p. 16, lines 14-24:

As an example, a model for a component that is an application, such as Microsoft® Word, can first be generated. Such a model will contain a collection of all of the data that define the presence, attributes, and dependencies of the components, such as the filenames and directory structure of the files that make up Microsoft® Word. Because the model for Microsoft® Word will contain a large amount of data, **a smaller subset of this data** will be compiled into a fingerprint that can be used for discovery purposes. The fingerprint for Microsoft® Word, for instance, could contain the key executable files and data the makeup and uniquely define Microsoft® Word.

p. 18, lines 24-28:

After a model of a component has been created, a fingerprint, or matching set, of the model can be constructed. This fingerprint, which is **a subset of the model** for a known component, can be used to discover the presence of an existing component in the IT system. A model for component, for instance, might contain hundreds of elements, but a fingerprint for the same component might contain only ten to twenty elements.

With respect to the claim limitations of, ". . .automatically discovering . . . using the plurality of component fingerprints . . . ", ". . .wherein each event message matches a respective attribute of the fingerprint . . ." and ". . .determining that event messages matching every attribute of the fingerprint . . .", see at least the following portions of the specification:

p. 14, lines 10-15:

The analysis service 70 can, in one embodiment, use fingerprints to analyze event information to determine if any of the components in the fingerprint data base 84 exist on the server 11 of the agent 12. As will be described in more detail below, the accumulator 80 can be used to **determine if all of the elements of the fingerprint exist**, which indicates the presence of the component indicated by the fingerprint.

p. 16, lines 4-9:

As an example of a model-based discovery, the agent 12 or network server 10 can use a fingerprint to discover that an existing component has been installed on the IT system through the use of the accumulation of real-time event information or by inspecting the actual contents of the IT system to see if the components are present that match the fingerprint of the model of a known

component. If components are present that match the fingerprint of the known component, the existing component on the IT system is discovered.

p. 16, lines 14-24:

As an example, a model for a component that is an application, such as Microsoft® Word, can first be generated. Such a model will contain a collection of all of the data that define the presence, attributes, and dependencies of the components, such as the filenames and directory structure of the files that make up Microsoft® Word. Because the model for Microsoft® Word will contain a large amount of data, a smaller subset of this data will be compiled into a fingerprint that can be used for discovery purposes. The fingerprint for Microsoft® Word, for instance, could contain the key executable files and data the makeup and uniquely define Microsoft® Word. During a discovery process, information about a number of events (that is, file or registry entry creations or deletions) can be accumulated that form parts of the fingerprint for Microsoft® Word. When the last of these of events is discovered, the fingerprint for Microsoft® Word has been matched and the existing Microsoft® Word component on the IT system has been discovered.

p. 30, lines 11-18

At block 204 of FIGURE 9, the fingerprints and subfingerprints are used to discover components in the IT system. Referring to FIGURE 2, the analysis service 70 of the agent 12, including the rule engine 74, accumulator 80, and fingerprint database 84 can be used for the discovery of components at the agent-level. The analysis at the agent level generally focuses on the local system of the agent or the remote server that the agent monitors. As described above, event information of a fingerprint from the fingerprint database 84 can be matched in the accumulator 80 until all of the passive elements of the fingerprint had been matched. At that point, an application discovered message can be generated by the fingerprint and the rule engine 74.

p. 8, lines 20-24:

The observer service 50 is responsible for loading and configuring a set of observers that can be used to collect event information. An observer is a piece of code that determines if an event has occurred that is significant to the IT system, generates a message describing the event, and passes the message to the agent 12 for further processing. Such events can be detected in real time as they occur…

p. 9, lines 3-5:

As noted above, the observers generate messages having event information that describe an event upon the occurrence or detection of the event. FIGURE 2 depicts the transmission of event information 94 in messages between the observer service 50 and the analysis service 70.

p. 14, lines 1-3:

> The analysis service 70 of the agent 12 of FIGURE 2 processes the event messages generated by the observer service 50 to detect components, track changes to components, and discover dependencies between components on the local agent 12.

Appellant thus respectfully submits that there is ample and clear support for the limitations of claims 5, 14, and 15 in the portions of the specification quoted above, as well as throughout the specification in general.

Claim 13

With respect to **claim 13**, the Examiner asserts:

> Independent claim 13 has been amended with the limitations of ". . .and listing one or more, but not all, of the plurality of attributes in a fingerprint . . ." and ". . .to determine that events matching every attributes listed in the fingerprint for . . ." that are not clearly specified in applicant's original specification or claim language. It would cause undue experimentation to one of ordinary skill in the art to make Applicant's invention

Appellant again disagrees. With respect to the claim limitation of, " listing one or more, but not all, of the plurality of attributes in a fingerprint," see at least the portions of the specification quoted above at p. 15, lines 24-29; p. 16, lines 14-24; and p. 18, lines 24-28.

With respect to the claim limitation of, "to determine that events matching every attributes listed in the fingerprint for . .", see at least the portions of the specification quoted above at p. 14, lines 10-15; p. 16, lines 4-9; p. 16, lines 14-24; p. 30, lines 11-18; p. 8, lines 20-24; p. 9, lines 3-5; and p. 14, lines 1-3.

Appellant thus respectfully submits that there is ample and clear support for the limitations of claim 13 in the cited sections of the specification, as well as throughout the specification in general.

Claims 16 and 23-24

With respect to **claims 16 and 23-24**, the Examiner asserts:

> Independent claims 16 and 23-24 have been amended with the limitations of

". . .creating a fingerprint . . . ", ". . . selecting one or more, but not all, of the plurality of attributes . . . ", ". . .creating a subfingerprint for a refinement of the first component … includes one or more attributes of the refinement of the first component . . . ", ". . .wherein each event message matches a respective attribute of the fingerprint . . ." and ". . .determining that event messages matching every attribute of the fingerprint. . ." that are not clearly specified in applicant's original specification or claim language. It would cause undue experimentation to one of ordinary skill in the art to make Applicant's invention.

Appellant again disagrees. With respect to the claim limitations of, "…creating a fingerprint . . . ", ". . . selecting one or more, but not all, of the plurality of attributes…", see at least the portions of the specification quoted above at p. 15, lines 24-29; p. 16, lines 14-24; and p. 18, lines 24-28.

With respect to the claim limitations of, ". . .wherein each event message matches a respective attribute of the fingerprint . . ." and ". . .determining that event messages matching every attribute of the fingerprint. . .", see at least the portions of the specification quoted above at p. 14, lines 10-15; p. 16, lines 4-9; p. 16, lines 14-24; p. 30, lines 11-18; p. 8, lines 20-24; p. 9, lines 3-5; and p. 14, lines 1-3.

With respect to the claim limitations of, "creating a subfingerprint for a refinement of the first component … includes one or more attributes of the refinement of the first component . . . ", see at least the following portions of the specification:

p. 2, lines 16-19:

Refined components, which are components that relate in some manner to another component (that is, the refined component is a specific version of the component or an optional piece that can be included with the component), can be discovered using subfingerprints that are activated upon the discovery of the component.

p. 19, lines 24-30:

A "subfingerprint" is a fingerprint that is used by the active elements of a parent fingerprint to discover a "refined component," which can be either a specific version of the component defined by the parent fingerprint or an optional piece that might be contained under the application defined by the parent fingerprint. Typically, a subfingerprint will contain information that is more refined than the information contained in a parent fingerprint. For example, a parent fingerprint might contain a list of file names to search for, and the subfingerprint might

contain not only file names, but the size of files as well. In addition, a subfingerprint can contain items that are not in the parent fingerprint.

p. 20, lines 7-11:

After all of the passive elements of the fingerprint Fl have been matched, the active elements of fingerprint F1 might cause a command message (FIGURES 2 and 3 illustrate commands 92 in transit), such as a message to retrieve more detailed information, to be sent to an observer to retrieve the size and checksum of file1.exe and then to attempt to match one or more subfingerprints.

p. 21, lines 4-6:

In such embodiments, the elements of the fingerprint in addition to the elements of the subfingerprint would have to be matched in order to discover the refined component of the subfingerprint.

p. 30, lines 18-23:

In some embodiments, the active elements of the fingerprint can then trigger command messages to search for certain types of elements, and subfingerprints can then be matched to discover subcomponents that relate in some way to the component of the original fingerprint (that is, versions of the component or optional pieces that can be used with the component). The subfingerprints can then be matched in the same manner as the fingerprints.

Appellant thus respectfully submits that there is ample and clear support for the limitations of claims 16 and 23-24 in the cited sections of the specification, as well as throughout the specification in general.

2) Section 102 Rejection (Kar) (Claims 2-5, and 11-15)

Claims 2-5 and 11-15 stand rejected under 35 U.S.C. 102(a) as being anticipated by Kar et al., "An Architecture for Managing Application Services over Global Networks" (hereinafter "Kar"). Appellant respectfully traverses this rejection.

Independent claims 5, 14, and 15

Claim 5 recites in pertinent part:

> creating a plurality of component fingerprints, wherein the plurality of component fingerprints includes a fingerprint for a first component, wherein creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component;
>
> automatically discovering the existence of a plurality of components in an information technology (IT) system using the plurality of component fingerprints, wherein the discovered components include the first component, wherein automatically discovering the existence of the first component comprises:
>
> receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component; and
>
> determining that event messages matching every attribute of the fingerprint for the first component have been received;

Kar

Kar does not even remotely teach this subject matter and is not even particularly relevant to this subject matter. Kar relates generally to a system for managing application services deployed on IP networks. Kar describes that network service providers (NSPs) already have well-developed network management infrastructures to operate their physical networks. Since NSPs have gradually moved beyond the network layer to deploy and offer shared application services, an important objective is to leverage their existing network management infrastructure and enhance it to provide application service management. Kar describes an architecture that addresses this point by extending existing network management infrastructures with application service management capabilities, i.e., an architecture that exploits network management functions to provide fault and performance management for networked application services. (See page 1, last paragraph of left column through second paragraph of right column).

Kar describes that the physical network is partitioned into monitored domains, each of which is under the supervision of a typically SNMP-based management system, called a mid level manager (MLM). (See page 2, first paragraph of left column). An application service depends on the performance and status of resources that belong to multiple monitored domains. Therefore, monitoring the health and status of an application service requires monitoring information across the physical domains in which these resources are located. To this end, Kar introduces the concept of service management domains which form the basis of information monitoring for managing fault, performance, and service level agreements related to application service offerings. (See page 2, second paragraph of left column).

Claim 5

Referring now to claim 5, Appellant notes that Kar's disclosure bears very little resemblance or relevance to the subject matter of the above-recited limitations. More particularly, Kar teaches nothing whatsoever about automatically discovering the existence of a first component using a fingerprint for the first component, as recited in claim 5. Appellant submits that Kar fails to teach a number of the limitations of claim 5.

"creating the first fingerprint for the first component…"

First, Kar does not teach creating a fingerprint for a first component, as recited in the following limitations of claim 5:

> creating a plurality of component fingerprints, wherein the plurality of component fingerprints includes a fingerprint for a first component, wherein creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component;

With respect to these limitations the Examiner cites at page 5 of the Office Action:

> (page 6, right column, item 1 : resource identifier; page 7, right column, 2nd paragraph: software definition file formats with many attributes, dependencies between components; page 2, left column, 2nd paragraph: distributed components, resources belong to multiple monitored domains; page 8, left column, last paragraph: application components)

As shown in the following discussion, Appellant submits that none of the passages cited by the Examiner teach or suggest the above-quoted limitation of claim 5.

As for the section of Kar cited at **page 6**, Kar teaches here that an Application Service Agent in charge of a particular application service sends a request to a Resource Broker, including a list containing the resource identifiers of all the resources that the application service is dependent on. The Resource Broker uses the resource identifiers to obtain a list of the mid level managers (MLMs) responsible for monitoring these resources. The Resource Broker contacts each of the MLMs and sends each MLM the Application Service Agent ID and the associated resource list. This informs each MLM that events related to the identified resources should be forwarded to the Application Service Agent. Thus, when an event related to a particular resource is generated, the MLM responsible for monitoring that resource is notified of the event, and the MLM then forwards the event information along with the Application Service Agent ID to the network management platform. The network management platform in turn forwards the event to the Application Service Agent. (See items 1 thru 7 in the right column of page 6 and the left column of page 7).

Thus, Kar's system enables an Application Service Agent in charge of a particular application service to receive events from a mid level manager (MLM) for an SNMP-based management system, where the events are relevant to particular resources used by the application service. This enables the Application Service Agent to monitor the status of the resources used by the application service. However, this teaches nothing whatsoever about the recited claim limitation of creating the fingerprint for the first component by identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component, where the fingerprint for the first component is used to automatically discover the existence of the first component.

As for the section of Kar cited at **page 7, right column, 2nd paragraph**, Kar simply teaches here a software definition file format with many attributes, which is used for software distribution/deployment/installation. This has no relevance at all to the recited limitation of creating the fingerprint for the first component by identifying a

plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component, where the fingerprint for the first component is used to automatically discover the existence of the first component.

As for the section of Kar cited at **page 2, left column, 2nd paragraph**, this section was already discussed above, and generally describes the problem with which Kar is concerned, i.e., monitoring the health and status of an application service by monitoring the lower level resources used by the application service. Again, this is not at all relevant to the recited claim limitations of creating the fingerprint for the first component by identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component, where the fingerprint for the first component is used to automatically discover the existence of the first component. Likewise, the cited section at **page 8, left column, last paragraph** teaches nothing at all about these claim limitations.

Appellant thus submits that Kar completely fails to teach the above-recited limitations regarding the creation of the fingerprint for the first component, in combination with the other limitations recited in claim 5. Appellant also notes that although the Examiner has cited broad portions of Kar's disclosure with respect to these limitations, it is not at all clear what exactly in Kar the Examiner considers to be the fingerprint for the first component. Appellant respectfully submits that there is nothing in Kar that could reasonably be interpreted as the fingerprint for the first component recited in claim 5.

"automatically discovering the existence of the first component…"

Second, Appellant also submits that Kar fails to teach the further limitations of:

> automatically discovering the existence of a plurality of components in an information technology (IT) system using the plurality of component fingerprints, wherein the discovered components include the first component, wherein automatically discovering the existence of the first component comprises:
>
> receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component; and
>
> determining that event messages matching every attribute of the fingerprint for the first component have been received;

As for the limitations of, "receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component," and "determining that event messages matching every attribute of the fingerprint for the first component have been received," the Examiner cites on pp. 5 and 6 of the Office Action:

> (page 2, right column, 2nd paragraph- page 3, left column, 1st paragraph: MLM; page 5, right column, last paragraph - page 6, left column, lst paragraph: Application Service Agent discovers resource through MLMs)

and

> (page 2, right column, 2nd paragraph- page 3, left column, lst paragraph: MLM polls and event notification to MLM; page 5, right column, last paragraph-page 6, left column, 1st paragraph: Application Service Agent discovers resource through MLMs);

As discussed above, it is not at all clear what in Kar the Examiner considers to be the fingerprint for the first component, and thus it is also not clear what in Kar the Examiner considers to be the "attributes" of the fingerprint for the first component, or how the Examiner considers Kar to teach receiving event massages that match the attributes of the fingerprint. As for the sections of Kar cited with respect to these limitations, the content of these sections was already described above. Kar teaches that when an event related to a particular resource used by an application service is generated, the MLM responsible for monitoring that resource is notified of the event. The MLM then forwards the event information along with the Application Service Agent ID to the network management platform. The network management platform in turn forwards the event to the Application Service Agent. This effectively allows the Application Service Agent which manages the application service to monitor the health of the lower-level resources used by the application service. Thus, Appellant disagrees with the Examiner's assertion quoted above that, "Application Service Agent discovers resource through MLMs". The events passed to the Application Service Agent are not used to discover resources as asserted by the Examiner. The Application Service Agent already knows the resources on which the application service depends, and the events passed to the

Application Service Agent simply allow the Application Service Agent to monitor the status of these resources.

Appellant thus respectfully submits that Kar does not even remotely teach the recited limitations of:

> wherein automatically discovering the existence of the first component comprises:
>
> receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component; and
>
> **determining that event messages matching every attribute of the fingerprint for the first component have been received**;

Appellant thus respectfully submits that claim 5 is patentably distinct over Kar for at least the reasons set forth above.

"tracking changes to at least one of the discovered components …"

Third, claim 5 recites the further limitation of, "tracking changes to at least one of the discovered components and the at least one dependency between two or more of the discovered components." With respect to this limitation, the Examiner cites on p. 6 of the Office Action:

> (page 2, right column, 2nd paragraph - page 3, left column, 1st paragraph: MLM; page 5, right column, 2nd paragraph-page 6, left column, 4th paragraph: Application Service Agents, Resource Broker and Resource Directory).

The cited sections generally relate to Kar's system as described above. However, Appellant can find no teaching either here or elsewhere in Kar of the specifically recited claim limitations of, "tracking changes to at least one of the discovered components and the at least one dependency between two or more of the discovered components." Appellant thus submits that claim 5 is also patentably distinct over Kar for this further reason.

Since claim 5 is patentably distinct over Kar, the claims dependent thereon are also patentably distinct for at least this reason. Inasmuch as the independent claims 14 and 15 recite similar limitations as claim 5, Appellant submits that these claims are also patentably distinct over Kar, for reasons similar to those discussed above.

Claim 13

Independent claim 13 recites limitations similar to those discussed above with respect to claim 5, such as:

> identifying a plurality of attributes of a first component and listing one or more, but not all, of the plurality of attributes in a fingerprint for the first component;
>
> an observer module detecting a plurality of real-time events in an information technology (IT) system, wherein each event matches a respective attribute listed in the fingerprint for the first component;
>
> the observer module sending a respective event message to an analysis module in response to each of the plurality of real-time events;
>
> the analysis module accumulating the event messages and analyzing the event messages in order to determine that events matching every attribute listed in the fingerprint for the first component have occurred in the IT system; and
>
> the analysis module indicating that the first component exists in the IT system in response to said determining that events matching every attribute listed in the fingerprint for the first component have occurred in the IT system.

The Examiner stated that claim 13 was rejected for the same reasons as claim 5. Thus, Appellant respectfully submits that the rejection of claim 13 is erroneous for reasons similar to those set forth above for claim 5.

Claim 12

Dependent claim 12 is dependent upon claim 5 and recites the further limitation of, "generating a component discovered message upon the discovery of one of the components". The Examiner asserts that this is taught by Kar at page 2, right column, 2nd paragraph, 1st-3rd bullets.

Appellant respectfully disagrees. The cited sections teach functions performed by a mid level manager (MLM), such as polling the SNMP agents in its domain, receiving events from the SNMP agents, and storing polled data in a database. There is nothing whatsoever in the cited section teaching "generating a component discovered message upon the discovery of one of the components." The MLM clearly already knows the resources that it manages. The cited portion does not pertain to the discovery of

components, but instead refers to the MLM receiving events regarding the resources managed by the MLM.

Appellant thus respectfully submits that claim 12 is separately patentable over Kar for at least this further reason.

3) Section 102 Rejections (Keller) (Claims 30, 32-33, and 35-40)

Claims 30, 32-33, and 35-40 stand rejected under 35 U.S.C. 102(a) as being anticipated by Keller et al., "Dynamic Dependencies in Application Service Management" (hereinafter "Keller"). Appellant respectfully traverses this rejection.

Claims 30 and 40

Independent claim 30 recites in pertinent part:

> automatically discovering a first component and a second component in the IT system, wherein automatically discovering the first component comprises automatically discovering that one or more elements of the first component are present in the IT system, wherein automatically discovering the second component comprises automatically discovering that one or more elements of the second component are present in the IT system;

First, Appellant notes that Keller relates generally to dependency analysis in an application service management system, e.g., analyzing dependencies between components in an application service management system. However, Keller does not teach automatically discovering the components for which the dependencies are analyzed. In particular, Keller does not teach "automatically discovering a first component" (or a second component) in an IT system, wherein automatically discovering the first component (or the second component) comprises automatically discovering that one or more elements of the first component (or the second component) are "present in the IT system." Appellant thus respectfully submits that claim 30 is patentably distinct over Keller for at least this reason.

Second, claim 30 also recites the further limitations of:

> monitoring the usage of resources by the discovered first and second components in the IT system by receiving real-time messages;
>
> in response to receiving a first real-time message indicating that the first component uses a particular resource, sending a first resource usage message to

an accumulator, wherein the first resource usage message indicates that the first component uses the particular resource;

in response to receiving a second real-time message indicating that the second component uses the particular resource, sending a second resource usage message to the accumulator, wherein the second resource usage message indicates that the second component uses the particular resource;

the accumulator indicating that a first dependency between the first component and the second component exists in response to determining that the first and second resource usage messages indicate that the first component and the second component both use the particular resource;

Appellant respectfully submits that Keller does not teach these limitations. Keller emphasizes repeatedly throughout its disclosure that dependencies are statically analyzed by analyzing information stored in repositories. See, for example, the following passages of Keller's disclosure:

> The approach for identifying and computing dependencies presented in this paper is pragmatic and based on a **static dependency analysis** that yields information on entities within a system (Intrasystem) and between peer entities of a service (Intersystem). [Section 6: Conclusion and Outlook, p. 7, right column, second paragraph]

> Considering the fact that a majority of application services run on UNIX and Windows NT-based systems, it is worth analyzing the degree to which information regarding applications and services is already contained in the operating systems. The underlying idea is as follows: if it is possible to obtain a reasonable amount of information from these sources, the need for application-specific instrumentation can be greatly reduced. Our approach recognizes the fact that system administrators successfully deploy applications and services without having access to detailed, application-specific management instrumentation.
>
> Windows NT/95/98 systems and UNIX implementations such as IBM AIX and Linux have built-in repositories that keep track of the installed software packages, filesets and their versions. AIX Object Data Manager (ODM), Windows Registry, and Linux Red Hat Package Manager (RPM) are examples for these system-wide configuration repositories. In this paper, we will concentrate on ODM. However, we have verified the applicability of our approach to the other repositories as well. [Section 4: Dependency Analysis, p. 5, bottom of left column to top of right column]

> Our analysis has shown that system repositories such as ODM represent a rich source of application service management information, not only regarding the configuration of installed applications but also for determining dependency

relationships between applications and services. Note that this large amount of information can be obtained *without any* specific instrumentation of the components. The only requirement is that the application components be described using a service description template that has been developed by us and whose content can be provided to a large degree by today's system information repositories. [Section 4: Dependency Analysis, p. 6, left column]

We assume that through **offline analysis – based on the methodology and approach presented in section 4** – a database of static dependencies is constructed. This collected data describes, for each end-to-end application service, the dependencies it has on lower level application and network layer services and components. [Section 5: Dependency Architecture, p. 6, right column]

From the passages quoted above, it is very clear that Keller's method for determining dependency relationships involves the static, offline analysis of system repositories. In contrast, the dependency between the first component and the second component in claim 30 is indicated by the accumulator in response to determining that the first and second resource usage messages indicate that the first component and the second component both use the particular resource, wherein the first resource usage message is sent to the accumulator in response to receiving a first real-time message indicating that the first component uses the particular resource, and wherein the second resource usage message is sent to the accumulator in response to receiving a second real-time message indicating that the second component uses the particular resource. Keller simply does not teach these limitations, let alone in combination with the other limitations recited in claim 30.

Appellant thus respectfully submits that claim 30, and the claims dependent thereon, are patentably distinct over Keller for at least the reasons set forth above. Inasmuch as claim 40 recites similar limitations as claim 30, Appellant respectfully submits that claim 40 is also patentably distinct over Keller.

Claim 33

Claim 33 is dependent upon claim 32 and recites the further limitations of:

> wherein the particular resource is a particular network port;
> wherein the first resource usage message indicates that the first component uses the particular network port and wherein the second resource

usage message indicates that the second component uses the particular network port.

In the rejection of claim 33 the Examiner cites Keller at "(Fig. 1; page 2, left column, 1st paragraph and 2nd paragraph, item 1; page 6, right column, lst paragraph: network layer services and components)." However, these portions of Keller, and Keller in general, do not teach the specifically recited limitations of receiving a first resource usage message indicating that the first component uses the particular network port and a second resource usage message indicating that the second component uses the particular network port.

Appellant thus respectfully submits that claim 33 is separately patentable over Keller for at least this reason.

Claim 35

Claim 35 is dependent upon claim 32 and recites the further limitations of:

> wherein the particular resource is a particular file;
> wherein the first resource usage message indicates that the first component uses the particular file and wherein the second resource usage message indicates that the second component uses the particular file.

In the rejection of claim 35 the Examiner cites Keller at "(Fig. 1; page 2, left column, 1st paragraph and 2nd paragraph, item 1; page 3, right column: Component Type and Component Activity)." However, these portions of Keller, and Keller in general, do not teach the specifically recited limitations of receiving a first resource usage message indicating that the first component uses the particular file and a second resource usage message indicating that the second component uses the particular file.

Appellant thus respectfully submits that claim 35 is separately patentable over Keller for at least this reason.

4) Section 103 Rejections (Kar in view of Kathrow) (Claims 6-10)

Claims 6-10 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Kar in view of Kathrow et al., U.S. Patent No. 6,393,438 B1 (hereinafter "Kathrow"). Appellant respectfully traverses this rejection.

Claim 6

Claim 6 is dependent upon claim 5 and recites the further limitation of:

> wherein the plurality of event messages includes a first event message indicating a first real-time event selected from the following real-time events: a file creation, a file deletion, and a file modification.

In the rejection of claim 6 the Examiner cites Kathrow at Col. 4, lines 60-65. Kathrow teaches here that a characteristic identifier identifies one or more characteristics of a file, such as the location of the file on the disk, last modification or update date of the file, or any other characteristic of the file. However, Kathrow does not teach receiving a real-time event message indicating a file creation, a file deletion, or a file modification. Thus, because the Examiner is relying on Kathrow to teach the limitations of claim 6 and Katrhow does not teach these limitations, the proposed combination of Kar and Kathrow would not include each and every limitation of claim 6. Thus, such a combination fails to establish a *prima facie* case of obviousness with respect to claim 6.

Furthermore, there is no apparent motivation to combine Kathrow's teaching of identifying file characteristics with Kar's system. As discussed above, the Examiner has equated the recited event messages with the event messages that Kar's application service agent receives from the MLMs. The event messages pertain to events regarding resources used by the application service which the application service agent manages.

In contrast, Kathrow teaches determining a hash of a file and a size of the file, which are two 4-byte values that are together referred to as a "fingerprint" for the file. The file fingerprint (also referred to as a signature – See Col. 10, lines 16-18 and Abstract) is used to identify the existence of differences between two files. The file characteristics identified by Kathrow as taught in the section cited by the Examiner are apparently used to create the fingerprint of the file. The problem of creating a file fingerprint as taught by Kathrow is not even remotely related to the problem addressed by

Kar's system, and there is no motivation to combine Kar's teaching of identifying file characteristics with Kar's teaching of an MLM forwarding event messages to the application service agent.

Claim 7

Claim 7 is dependent on claim 5 and recites the further limitation of:

> wherein the plurality of event messages includes a first event message indicating a first real-time event selected from the following real-time events: a registry key creation, a registry key deletion, and a registry key modification.

In the rejection of claim 7 the Examiner cites Kathrow at Col. 4, line 60-Col. 5, line 4. Kathrow teaches here that:

> In an alternative embodiment of the present invention, characteristic identifier 224 identifies one or more other characteristics of the file either in place of the size or in addition to the size. Such characteristics can include the location of the file on the disk, last modification or update date of the file, or any other characteristic of the file.
>
> Extractor 226 extracts the values, the records containing the keys and values, or fixed length blocks from the Windows registry file stored in file storage 210. In the description that follows, extractor 226 is described as extracting only values, although these other portions or any other type of portion may be extracted by extractor 226.

Thus, Kathrow simply teaches here that the extractor extracts information stored in the Windows registry file. However, Kathrow, taken either singly or in combination with Kar, does not teach receiving a first event message indicating a first real-time event selected from the following real-time events: a registry key creation, a registry key deletion, and a registry key modification. Thus, because the Examiner is relying on Kathrow to teach the limitations of claim 7 and Katrhow does not teach these limitations, the proposed combination of Kar and Kathrow would not include each and every limitation of claim 7. Thus, such a combination fails to establish a *prima facie* case of obviousness with respect to claim 7.

Furthermore, there is no apparent motivation to combine Kathrow's teaching of extracting registry information with Kar's system, for reasons similar to those discussed above with reference to claim 6.

Claim 8

Claim 8 is dependent on claim 5 and recites the further limitation of:

> wherein the plurality of event messages includes a first event message indicating detection of a particular element of the first component in the IT system.

In the rejection of claim 8 the Examiner cites Kathrow at Col. 4, line 60- Col. 5, line 26. The cited portion of Kathrow relates generally to the creation of a file signature used to identify the existence of differences between files. However, Kathrow teaches nothing about a first event message indicating detection of a particular element of the first component in the IT system, as recited in claim 8. Because the Examiner is relying on Kathrow to teach the limitations of claim 8 and Katrhow does not teach these limitations, the proposed combination of Kar and Kathrow would not include each and every limitation of claim 8. Thus, such a combination fails to establish a *prima facie* case of obviousness with respect to claim 8.

Furthermore, there is no apparent motivation to combine Kathrow's teaching of identifying the existence of differences between files with Kar's system, for reasons similar to those discussed above with reference to claim 6.

Claims 9 and 10

Claim 9 is dependent on claim 5 and recites the further limitations of:

> after discovering the existence of the first component, receiving a subsequent event message indicating that an element of the first component was deleted; and
>
> indicating that the first component has been damaged in response to the subsequent event message.

In the rejection of claims 9 and 10 the Examiner cites the following portions of Kathrow:

> In an alternative embodiment of the present invention, characteristic identifier 224 identifies one or more other characteristics of the file either in place of the size or in addition to the size. Such characteristics can include the location of the

> file on the disk, last modification or update date of the file, or any other characteristic of the file. (Col. 4, lines 60-65)
>
> In one embodiment, step 440 also includes notifying the user whether the fingerprints were identical. If the fingerprints are not identical 434, differences may be identified 436 between the values corresponding to the two fingerprints compared in step 432. (Col. 11, lines 62-64)
>
> If differences exist, corresponding individual hash results from each signature are compared to identify differences in step 436. The blocks stored in step 420 that correspond to the individual hash results identified in step 436 may be received from the known good file or from the blocks optionally stored in step 420 and stored in the other file or file version as part of step 438 to match the other file or version with the known good file or version. (Col. 14, lines 8-15)

These portions relate generally to the use of file signatures to identify differences between files, as taught by Kathrow. However, Kathrow, taken either singly or in combination with Kar, does not teach indicating that the first component has been damaged in response to receiving a subsequent event message indicating that an element of the first component was deleted, as recited in claim 9. Because the Examiner is relying on Kathrow to teach the limitations of claim 9 and Katrhow does not teach these limitations, the proposed combination of Kar and Kathrow would not include each and every limitation of claim 9. Thus, such a combination fails to establish a *prima facie* case of obviousness with respect to claim 9.

Likewise, Kathrow, taken either singly or in combination with Kar, does not teach indicating that the first component has been uninstalled in response to receiving one or more subsequent event messages indicating that one or more elements of the first component were deleted, as recited in claim 10. Because the Examiner is relying on Kathrow to teach the limitations of claim 10 and Katrhow does not teach these limitations, the proposed combination of Kar and Kathrow would not include each and every limitation of claim 10. Thus, such a combination fails to establish a *prima facie* case of obviousness with respect to claim 10.

Furthermore, there is no apparent motivation to combine Kathrow's teaching with Kar's system, for reasons similar to those discussed above with reference to claim 6.

5) Section 103 Rejections (Kar in view of Kathrow) (Claims 16-18 and 22-24)

Claims 16-18, and 22-24 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Kar in view of Kathrow et al., U.S. Patent No. 6,393,438 B1 (hereinafter "Kathrow"). Appellant respectfully traverses this rejection.

Claims 16, 23, and 24

Claim 16 recites:

> 16. A method comprising:
>
> creating a fingerprint for a first component, wherein creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component;
>
> creating a subfingerprint for a refinement of the first component, wherein the subfingerprint for the refinement of the first component includes one or more attributes of the refinement of the first component;
>
> automatically discovering the first component in an information technology (IT) system, wherein automatically discovering the first component comprises:
>
> receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component; and
>
> determining that event messages matching every attribute of the fingerprint for the first component have been received;
>
> wherein the method further comprises:
>
> in response to discovering the first component, performing one or more commands to obtain information regarding the first component; and
>
> automatically discovering the refinement of the first component in the IT system by matching the information regarding the first component to the one or more attributes of the refinement of the first component included in the subfingerprint for the refinement of the first component.

Claim 16 includes limitations similar to claim 5, such as the limitations regarding creating the first fingerprint for the first component, and automatically discovering the first component by receiving a plurality of event messages matching respective attributes

of the fingerprint for the first component, and determining that event messages matching every attribute of the fingerprint for the first component have been received. The Examiner relies largely on Kar to teach these limitations. However, as discussed in detail above with reference to claim 5, Kar utterly fails to teach, or even bear much relevance to, these limitations. Thus, to the extent that the rejection of claim 16 relies largely on Kar, Appellant submits that the rejection is defective for at least this reason.

Furthermore, Appellant also submits that the combination of Kathrow with Kar does not teach the features asserted by the Examiner. The Examiner cites Kathrow's teaching regarding the use of file fingerprints or signatures. Kathrow teaches determining a hash of a file and a size of the file, which are two 4-byte values that are together referred to as a "fingerprint" for the file. However, Kathrow, taken either singly or in combination with Kar, does not teach the limitations recited in claim 5 of:

> creating a subfingerprint for a refinement of the first component, wherein the subfingerprint for the refinement of the first component includes one or more attributes of the refinement of the first component;
>
> and
>
> automatically discovering the refinement of the first component in the IT system by matching the information regarding the first component to the one or more attributes of the refinement of the first component included in the subfingerprint for the refinement of the first component.

Kathrow teaches the use of a file fingerprint (also referred to as a signature – See Col. 10, lines 16-18 and Abstract) to identify the existence of differences between two files. However, Kathrow does not teach or even remotely suggest the creation of a subfingerprint for a refinement of a first component, where the subfingerprint for the refinement of the first component includes one or more attributes of the refinement of the first component. Kathrow (taken either singly or in combination with Kar) also fails to teach automatically discovering the refinement of the first component in the IT system by matching the information regarding the first component to the one or more attributes of the refinement of the first component included in the subfingerprint for the refinement of the first component.

Furthermore, there is no apparent motivation to combine Kathrow's invention with Kar's system. Kar is not concerned with the problem of identifying the existence of

differences between files. As discussed above, Kar discloses a system for monitoring the health and status of an application service by monitoring lower-level resources upon which the application service. It is difficult to see how Kathrow's teaching regarding the use of fingerprints to identify differences between files is particularly relevant to Kar's system. Appellant submits that the Examiner has not established a clear motivation to combine these references.

Appellant thus respectfully submits that claim 16, and the claims dependent thereon, are patentably distinct over the cited references for at least the reasons set forth above. Inasmuch as the independent claims 23 and 24 recite similar limitations as claim 16, Appellant submits that these claims are also patentably distinct over the cited references, for reasons similar to those discussed above.

Claim 22

Claim 22 is dependent on claim 16 and recites the further limitations of:

> wherein the plurality of event messages include one or more event messages indicating one or more real-time events associated with one or more of files, registry settings, and database schemas.

In the rejection of claim 22 the Examiner cites Kathrow at Col. 3, lines 59-62, which teaches:

> Referring now to FIG. 2, an apparatus for determining whether a two personal computer files, such as two versions of a Windows registry file, contain differences is shown according to one embodiment of the present invention.

This generally relates to Kathrow's invention for determining whether two files contain differences. In contrast, the plurality of event messages recited in claim 16 are used to automatically discover the existence of a first component in an IT system. In particular, claim 16 recites that the plurality of event messages include one or more event messages indicating one or more real-time events associated with one or more of files, registry settings, and database schemas. Kathrow, taken either singly or in combination with Kar, does not teach this subject matter.

VIII. CLAIMS APPENDIX

The following lists the claims involved in the appeal.

2. The method of claim 5, further comprising generating a visual map of the IT system, the visual map including a depiction of at least one of the discovered components and the at least one dependency between two or more of the discovered components.

3. The method of claim 2, wherein the visual map includes tracked changes to at least one of the discovered components.

4. The method of claim 5, wherein at least one of the discovered components is an application.

5. A method comprising:

creating a plurality of component fingerprints, wherein the plurality of component fingerprints includes a fingerprint for a first component, wherein creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component;

automatically discovering the existence of a plurality of components in an information technology (IT) system using the plurality of component fingerprints, wherein the discovered components include the first component, wherein automatically discovering the existence of the first component comprises:

receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component; and

determining that event messages matching every attribute of the fingerprint for the first component have been received;

wherein the method further comprises:

automatically determining at least one dependency between two or more of the discovered components; and

tracking changes to at least one of the discovered components and the at least one dependency between two or more of the discovered components.

6. The method of claim 5, wherein the plurality of event messages includes a first event message indicating a first real-time event selected from the following real-time events: a file creation, a file deletion, and a file modification.

7. The method of claim 5, wherein the plurality of event messages includes a first event message indicating a first real-time event selected from the following real-time events: a registry key creation, a registry key deletion, and a registry key modification.

8. The method of claim 5, wherein the plurality of event messages includes a first event message indicating detection of a particular element of the first component in the IT system.

9. The method of claim 5, further comprising:

after discovering the existence of the first component, receiving a subsequent event message indicating that an element of the first component was deleted; and

indicating that the first component has been damaged in response to the subsequent event message.

10. The method of claim 5, further comprising:

after discovering the existence of the first component, receiving one or more subsequent event messages indicating that one or more elements of the first component were deleted; and

indicating that the first component has been uninstalled in response to the one or more subsequent event messages.

11. The method of claim 5, wherein the at least one dependency is selected from the group consisting of shared library usage, network usage, and containment dependencies.

12. The method of claim 5, further comprising:

generating a component discovered message upon the discovery of one of the components;

retrieving a list of elements to track the discovered component; and using the list of elements to track changes to the discovered component.

13. A computer-readable medium storing program instructions that are computer executable to implement a method comprising:

identifying a plurality of attributes of a first component and listing one or more, but not all, of the plurality of attributes in a fingerprint for the first component;

an observer module detecting a plurality of real-time events in an information technology (IT) system, wherein each event matches a respective attribute listed in the fingerprint for the first component;

the observer module sending a respective event message to an analysis module in response to each of the plurality of real-time events;

the analysis module accumulating the event messages and analyzing the event messages in order to determine that events matching every attribute listed in the fingerprint for the first component have occurred in the IT system; and

the analysis module indicating that the first component exists in the IT system in response to said determining that events matching every attribute listed in the fingerprint for the first component have occurred in the IT system.

14. A system comprising:

means for creating a plurality of component fingerprints, wherein the plurality of component fingerprints includes a fingerprint for a first component, wherein creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component;

means for automatically discovering the existence of a plurality of components in an information technology (IT) system using the plurality of component fingerprints, wherein the discovered components include the first component, wherein automatically discovering the existence of the first component comprises:

receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component; and

determining that event messages matching every attribute of the fingerprint for the first component have been received;

means for automatically determining at least one dependency between two or more of the discovered components; and

means for tracking changes to at least one of the discovered components and the at least one dependency between two or more of the discovered components.

15. An apparatus comprising:

a memory storing program instructions;

a processor in communication with the memory; in which the processor is directed by the program instructions to:

create a plurality of component fingerprints, wherein the plurality of component fingerprints includes a fingerprint for a first component, wherein creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component;

automatically discover the existence of a plurality of components in an information technology (IT) system using the plurality of component fingerprints, wherein the discovered components include the first component, wherein automatically discovering the existence of the first component comprises:

receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component; and

determining that event messages matching every attribute of the fingerprint for the first component have been received;

automatically determine at least one dependency between two or more of the discovered components; and

track changes to at least one of the discovered components and the at least one dependency between two or more of the discovered components.

16. A method comprising:

creating a fingerprint for a first component, wherein creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component;

creating a subfingerprint for a refinement of the first component, wherein the subfingerprint for the refinement of the first component includes one or more attributes of the refinement of the first component;

automatically discovering the first component in an information technology (IT) system, wherein automatically discovering the first component comprises:

receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component; and

determining that event messages matching every attribute of the fingerprint for the first component have been received;

wherein the method further comprises:

in response to discovering the first component, performing one or more commands to obtain information regarding the first component; and

automatically discovering the refinement of the first component in the IT system by matching the information regarding the first component to the one or more attributes of the refinement of the first component included in the subfingerprint for the refinement of the first component.

17. The method of claim 16, wherein the refinement of the first component is a particular version of the first component, wherein discovering the refinement of the first component comprises discovering that the particular version of the first component exists in the IT system.

18. The method of claim 16, wherein the refinement of the first component is an optional piece of the first component, wherein discovering the refinement of the

first component comprises discovering that the optional piece of the first component exists in the IT system.

22. The method of claim 16, wherein the plurality of event messages include one or more event messages indicating one or more real-time events associated with one or more of files, registry settings, and database schemas.

23. A computer-readable medium storing instructions that direct a microprocessor to:

create a fingerprint for a first component, wherein creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting oen or more, but not all, of the plurality of attributes as the fingerprint for the first component;

create a subfingerprint for a refinement of the first component, wherein the subfingerprint for the refinement of the first component includes one or more attributes of the refinement of the first component;

automatically discover the first component in an information technology (IT) system, wherein automatically discovering the first component comprises:

receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component; and

determining that event messages matching every attribute of the fingerprint for the first component have been received;

in response to discovering the first component, perform one or more commands to obtain information regarding the first component; and

automatically discover the refinement of the first component in the IT system by matching the information regarding the first component to the one or more attributes of the refinement of the first component included in the subfingerprint for the refinement of the first component.

24. An apparatus comprising:

a memory storing program instructions;

a processor in communication with the memory; in which the processor is directed by the program instructions to:

create a fingerprint for a first component, wherein creating the fingerprint for the first component comprises identifying a plurality of attributes of the first component and selecting one or more, but not all, of the plurality of attributes as the fingerprint for the first component;

create a subfingerprint for a refinement of the first component, wherein the subfingerprint for the refinement of the first component includes one or more attributes of the refinement of the first component;

automatically discover the first component in an information technology (IT) system, wherein automatically discovering the first component comprises:

receiving a plurality of event messages indicating a plurality of real-time events that occur in the IT system, wherein each event message matches a respective attribute of the fingerprint for the first component; and

determining that event messages matching every attribute of the fingerprint for the first component have been received;

in response to discovering the first component, perform one or more commands to obtain information regarding the first component; and

automatically discover the refinement of the first component in the IT system by matching the information regarding the first component to the one or more attributes of the refinement of the first component included in the subfingerprint for the refinement of the first component.

30. A method for determining dependencies between components in an information technology (IT) system, comprising:

automatically discovering a first component and a second component in the IT system, wherein automatically discovering the first component comprises automatically discovering that one or more elements of the first component are present in the IT system, wherein automatically discovering the second component comprises automatically discovering that one or more elements of the second component are present in the IT system;

monitoring the usage of resources by the discovered first and second components in the IT system by receiving real-time messages;

in response to receiving a first real-time message indicating that the first component uses a particular resource, sending a first resource usage message to an accumulator, wherein the first resource usage message indicates that the first component uses the particular resource;

in response to receiving a second real-time message indicating that the second component uses the particular resource, sending a second resource usage message to the accumulator, wherein the second resource usage message indicates that the second component uses the particular resource;

the first component and the second component exists in response to determining that the first and second resource usage messages indicate that the first component and the second component both use the particular resource; and

determining a type of the first dependency between the first component and the second component.

32. The method of claim 30, wherein the first component is selected from the group consisting of an application, a network connection endpoint, and a server.

33. The method of claim 32,

wherein the particular resource is a particular network port;

wherein the first resource usage message indicates that the first component uses the particular network port and wherein the second resource usage message indicates that the second component uses the particular network port.

35. The method of claim 32,

wherein the particular resource is a particular file;

wherein the first resource usage message indicates that the first component uses the particular file and wherein the second resource usage message indicates that the second component uses the particular file.

36. The method of claim 30, further comprising tracking changes to the first dependency between the first component and the second component.

37. The method of claim 30, wherein determining the type of the first dependency comprises determining that the first dependency is a containment dependency.

38. The method of claim 30, wherein determining the type of the first dependency comprises determining that the first dependency is a network dependency.

39. The method of claim 30, wherein determining the type of the first dependency comprises determining that the first dependency is a shared usage dependency.

40. An apparatus comprising:

a memory storing program instructions;

a processor in communication with the memory; in which the processor is directed by the program instructions to implement a method comprising:

automatically discovering a first component and a second component in an information technology (IT) system, wherein automatically discovering the first

component comprises automatically discovering that one or more elements of the first component are present in the IT system, wherein automatically discovering the second component comprises automatically discovering that one or more elements of the second component are present in the IT system;

monitor the usage of resources by the discovered first and second components in the IT system by receiving real-time messages;

in response to receiving a first real-time message indicating that the first component uses a particular resource, sending a first resource usage message to an accumulator, wherein the first resource usage message indicates that the first component uses the particular resource;

in response to receiving a second real-time message indicating that the second component uses the particular resource, sending a second resource usage message to the accumulator, wherein the second resource usage message indicates that the second component uses the particular resource;

the first component and the second component exists in response to determining that the first and second resource usage messages indicate that the first component and the second component both use the particular resource; and

determining a type of the dependency between the first component and the second component.

IX. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

X. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.

XI. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 2-18, 22-24, 30, 32-33, and 35-40 was erroneous, and reversal of the decision is respectfully requested.

The fee of $510.00 for filing this Appeal Brief is being paid concurrently via EFS-Web. If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Appellant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/6002-07000.

Respectfully submitted,

Date: July 10, 2008

By: /Dean M. Munyon/
Dean M. Munyon
Reg. No. 42,914

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P. O. Box 398
Austin, Texas 78767
(512) 853-8847